



## 저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

이학석사 학위논문

Simulating Problem Difficulty in Arithmetic  
Cognition Through Dynamic Connectionist  
Models

동적 연결주의 모형을 통한 산술 인지 난이도 모사

2019 년 8 월

서울대학교 대학원  
협동과정 인지과학전공

조 성 재

# Simulating Problem Difficulty in Arithmetic Cognition Through Dynamic Connectionist Models

동적 연결주의 모형을 통한 산술 인지 난이도 모사

지도교수 장 병 탁

이 논문을 이학석사 학위논문으로 제출함

2019 년 8 월




서울대학교 대학원

협동과정 인지과학전공

조 성 재

조성재의 이학석사 학위논문을 인준함

2019 년 7 월

위 원 장	_____	고 성 룡	_____	(인) 
부위원장	_____	장 병 탁	_____	(인) 
위 원	_____	정 민 화	_____	(인) 

## **Abstract**

# Simulating Problem Difficulty in Arithmetic Cognition Through Dynamic Connectionist Models

Sungjae Cho

Interdisciplinary Program in Cognitive Science

The Graduate School

Seoul National University

The present study aims to investigate similarities between how humans and connectionist models experience difficulty in addition and subtraction problems. Problem difficulty was operationalized by the number of carries involved in solving a given problem. I aimed to simulate this human arithmetic cognition, performing either addition or subtraction, by using the Jordan network, which is a connectionist model dynamically computing outputs through time. The Jordan network is a recurrent neural network whose hidden layer gets its inputs from an input at the current step and from the output at the previous step. Problem difficulty was measured in humans by response time, and in models by computational steps. The present study found that both humans and connectionist models experience difficulty similarly when solving binary addition and subtraction. Specifically, both agents found difficulty to be strictly increasing with respect to the number of carries. Furthermore, the models mimicked the increasing standard deviation of response time seen in humans. Another notable similarity is that problem difficulty increases more steeply in subtraction than

in addition, for both humans and connectionist models. Further investigation on two model hyperparameters — confidence threshold and hidden dimension — shows higher confidence thresholds cause the model to take more computational steps to arrive at the correct answer. Likewise, larger hidden dimensions cause the model to take more computational steps to correctly answer arithmetic problems; however, this effect by hidden dimensions is negligible.

**Keywords:** arithmetic cognition; problem difficulty; response time; connectionist model; recurrent neural network; Jordan network; answer step

**Student Number:** 2017-28413

# Contents

<b>Abstract</b>	<b>i</b>
<b>Contents</b>	<b>iv</b>
<b>List of Tables</b>	<b>v</b>
<b>List of Figures</b>	<b>vi</b>
<b>Chapter 1 Introduction</b>	<b>1</b>
<b>Chapter 2 Problem Sets</b>	<b>8</b>
2.1 Operation Datasets . . . . .	8
2.2 Carry Datasets . . . . .	9
<b>Chapter 3 Experiment 1: Humans</b>	<b>10</b>
3.1 Participants . . . . .	10
3.2 Materials . . . . .	10
3.3 Procedure and Instruments . . . . .	11
3.4 Results . . . . .	13
3.4.1 Addition . . . . .	13
3.4.2 Subtraction . . . . .	13
<b>Chapter 4 Experiment 2: Connectionist Models</b>	<b>15</b>
4.1 Model . . . . .	15

4.2	Measures . . . . .	18
4.2.1	Accuracy . . . . .	18
4.2.2	Answer Step . . . . .	18
4.3	Training Settings . . . . .	20
4.4	Results . . . . .	20
4.4.1	Addition . . . . .	21
4.4.2	Subtraction . . . . .	22
<b>Chapter 5 Discussion and Conclusion</b>		<b>27</b>
<b>References</b>		<b>31</b>
<b>국문초록</b>		<b>35</b>

# List of Tables

Table 3.1	Means (and standard deviations) of mean RTs in Experiment 1 . . . . .	14
Table 4.1	Means (and standard deviations) of mean answer steps in Experiment 2 . . . . .	24
Table 4.2	The results of ANOVA and post hoc analysis on differences in mean answer steps between all carry datasets . .	25
Table 4.3	The results of ANOVA and post hoc analysis on differences in mean answer steps between confidence thresholds	26
Table 4.4	The results of ANOVA and post hoc analysis on differences in mean answer steps between hidden dimensions .	26



# List of Figures

Figure 1.1	Experimental phase diagram . . . . .	6
Figure 2.1	Problem sets . . . . .	8
Figure 3.1	Guiding examples . . . . .	12
Figure 3.2	Procedure of solving a problem in Experiment 1 . . . . .	12
Figure 3.3	Mean RT by carries . . . . .	14
Figure 4.1	The Jordan network used in the present study . . . . .	19
Figure 4.2	Mean answer step by carries (for carry datasets) . . . . .	23
Figure 4.3	Mean answer step by confidence threshold (for operation datasets) . . . . .	23
Figure 4.4	Mean answer step by hidden dimension (for operation datasets) . . . . .	23

# Chapter 1

## Introduction

Do connectionist models experience difficulty on arithmetic problems like humans? Although connectionist models consist of abstract biological neurons, similar behaviors between humans and these models are not guaranteed. However, developing model simulations to discover such similarities can bridge this knowledge gap between humans and models, and deepen our understanding of the micro-structures involved in cognition (Rumelhart & McClelland, 1986; McClelland, 1988). Therefore, finding such similarities is a foundational step in understanding human cognition through connectionist models. This connectionist approach recently has been used in the domain of mathematical cognition (Chen, Zhou, Fang, & McClelland, 2018; Fang, Zhou, Chen, & McClelland, 2018; Kuefler, Kochenderfer, & McClelland, 2017; McClelland, Mickey, Hansen, Yuan, & Lu, 2016; Mickey & McClelland, 2014; Saxton, Grefenstette, Hill, & Kohli, 2019).

Cognitive arithmetic (Ashcraft, 1992, 1995), the study of the mental representation of arithmetic, conceptualizes *problem difficulty*. Problem difficulty can be measured by *response time* (RT) from the time a participant sees an arithmetic problem to the time the participant answers the problem (Imbo, Vandierendonck, & Vergauwe, 2007).

There are three criteria that affect problem difficulty (Ashcraft, 1992, 1995):

(a) operand magnitude (e.g.,  $1 + 1$  vs.  $8 + 8$ ); (b) number of digits in the operands (e.g.,  $3 + 7$  vs.  $34 + 78$ ); and (c) the presence or absence of carry<sup>1</sup> operations (e.g.,  $15 + 31$  vs.  $19 + 37$ ). In particular, criterion (c) has been further investigated with regard to the number of carries required to correctly solve a problem (Fürst & Hitch, 2000; Imbo, Vandierendonck, & Vergauwe, 2007; Imbo, Vandierendonck, & De Rammelaere, 2007). In the present study, I investigated how the number of carries affected problem difficulty. *Response time* (RT) from the time a participant sees a problem to the time the participant answers the problem was used in the present study to measure problem difficulty.

Most studies compared problem difficulty between no-carry and one-carry problems. In contrast, the following studies found clear evidence that the number of carries in a problem affect both human accuracy and RT. Imbo, Vandierendonck, and Vergauwe (2007) investigated carry operations in subtraction between two 4-digit positive decimal numbers, and multiplication between a single-digit and a 3-digit positive decimal number. This study experimentally proved that the number of carry operations increased problem difficulty for both subtraction and multiplication. This study also found that executive working memory (Baddeley & Hitch, 1974; Baddeley & Della Sala, 1996) was used to perform carry operations fast and correctly. Another study by Imbo, Vandierendonck, and De Rammelaere (2007) examined carry operations in addition between four 4-digit positive numbers. This work found that more carries involved in addition problems resulted in increased problem difficulty, and that executive working memory was needed to perform carry operations fast and correctly.

Previous studies that examine the ways humans process numbers are mostly

---

<sup>1</sup>A *carry* in binary addition is the leading digit 1 shifted from one column to a more significant column when the sum of the less significant column exceeds a single digit. A *borrow* in binary subtraction is the digit  $10_{(2)} = 2$  shifted to a less significant column in order to obtain a positive difference in that column. This paper refers to borrows as carries.

based on the highly familiar decimal numeral system. Instead, the present study used the binary numeral system, which may offer a novel way to mitigate against the effect of previous experience with conventional mathematical operations. Moreover, since the binary system uses only 0 or 1 digits, it may reduce the *problem size effect*; criterion (a): problems with smaller operands (e.g.,  $5 + 2$ ,  $4 - 1$ ) are solved more quickly and accurately than problems with larger operands (e.g.,  $7 + 6$ ,  $9 - 6$ ) (Campbell, 1994; LeFevre et al., 1996; Miller, Perlmuter, & Keating, 1984). A previous study (Klein et al., 2010) has shown that the increasing effect of carry operations on RT is stronger for larger operands. Therefore, to observe the effect of carries on problem difficulty, the present study employed the binary system to control for familiarity with the decimal system and criterion (a).

When it comes to computational modeling, at least three different types of number representation have been studied in numerical cognition (Zorzi, Stoianov, & Umiltà, 2005): symbolic, number-line, and numerosity representations. *Symbolic* representation encodes each number into the activation of a dedicated node (e.g., (0, 1, 0, 0, 0, 0, 0, 0, 0, 0) for 1 and (0, 0, 0, 0, 1, 0, 0, 0, 0, 0) for 5). This approach views each number as a unique symbol that is orthogonal to all other numbers. Unlike the other two types of representations, symbolic representation does not include magnitude information. *Number-line* representation encodes each number into the activation of the corresponding node and its two adjoining neighbors (e.g., (.5, 1, .5, 0, 0, 0, 0, 0, 0, 0) for 1 and (0, 0, 0, .5, 1, .5, 0, 0, 0, 0) for 5). Number-line representation is based on the number-line hypothesis that suggests number magnitude is represented on a left-to-right oriented mental number line. As such, a number is encoded with activated points around the corresponding point on the number line. *Numerosity* representation straightforwardly encodes each number into the number of

activated units (e.g., (1, 1, 0, 0, 0, 0, 0, 0, 0) for 1 and (1, 1, 1, 1, 1, 0, 0, 0, 0) for 5).

The choice of representation can heavily influence the success or failure of a model (Bengio, Courville, & Vincent, 2013). The binary numeral system allows us to less consider the representation of numbers because the binary system yields only one (or few) representation (0 for 0 and 1 for 1) despite following the three preceding types of representations. For this reason, the binary system could help focus on analyzing the effect of carry operations on problem difficulty, independent of any influence from the choice of number representation. Therefore, the present study took advantage of the binary numeral system for connectionist models as well.

Extending the connectionist approach (Rumelhart & McClelland, 1986) to address problems of mathematical cognition could provide answers for a long lasting question whether neural networks can really think and reason as humans do, and further may help us understand in detail why mathematics is hard (McClelland et al., 2016). This approach is effective because connectionist models are able to learn many aspects of mathematical cognition. Also, these models offer the possibility to provide concrete instantiations of the mechanisms that grasp the nature of human knowledge and learning within the domain of mathematics.

Previous studies have demonstrated how connectionist models can simulate arithmetic operations. For instance, Anderson, Spoehr, and Bennett (2004); McCloskey and Lindemann (1992); Viscuso, Anderson, and Spoehr (1989) proposed associative-memory neural networks that stores a set of patterns representing single-digit multiplication operations. However, these networks were unable to learn all the given arithmetic operations. Franco and Cannas (1998) designed multilayer perceptrons (MLPs) that computed either the addition or

multiplication of two binary numbers. The MLPs were constructed with at least one hidden layer and binary step functions as activations. Instead of being learned from data, the weights of the MLPs above were analytically designed. Hoshen and Peleg (2016) made MLPs that learned arithmetic addition, subtraction and multiplication from images of two 7-digit decimal integers through a numerical method. Utilizing recent advances in deep learning (LeCun, Bengio, & Hinton, 2015), Kaiser and Sutskever (2016) implemented a convolutional gated recurrent network capable of learning either addition or multiplication of up-to 2000-bit binary numbers, trained on 20-bit numbers. This model achieved 100% test accuracy. However, the authors had to train 729 models with different random seeds to find one that attained 100% test accuracy. Notably, Mickey and McClelland (2014) demonstrated a deterministic recurrent neural network capable of filling blanks in 6 types of addition equations:  $a + b = \_$ ,  $a + \_ = b$ ,  $\_ + a = b$ ,  $a = b + \_$ ,  $a = \_ + b$ , and  $\_ = a + b$ . Addends and sums in these equations ranged from 0 to 9, and were represented as the number of active units in order to implement numerosity. This network simulated the strategies underlying the U-shape in child’s understanding (McNeil, 2007), which suggests that educators consider teaching more equations of various structure in their curricula. This work is relevant to the extend that it deals with problems of mathematical cognition through a connectionist approach (McClelland et al., 2016; Rumelhart & McClelland, 1986), and helps us understand which equivalence problems are hard. These previous studies were mostly centered around either cognitive psychology or artificial intelligence (AI). Cognitive psychologists tried to implement connectionist models to explain human cognition through simulation. Conversely, AI researchers tested their cutting-edge connectionist models by trying to achieve performance comparable to modern digital computers.

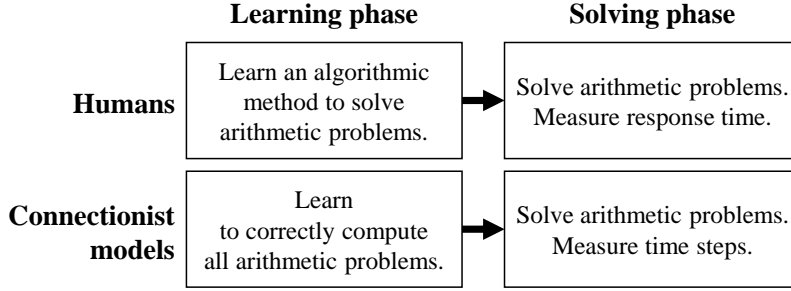


Figure 1.1: Experimental phase diagram

Recurrent neural networks (Elman, 1990; Jordan, 1997) can model sequential decisions through time. These networks perform sequential nonlinear computations. Owing to the principle that many nonlinear computational steps are required to learn complex mappings (LeCun et al., 2015), parallels can be drawn between human RT and model computational steps in response to problems of varying difficulty level. The present study simulated RT to solve arithmetic problems by employing the Jordan network (Jordan, 1997). To the best of my knowledge, the present study is the first to use a simple recurrent neural network to simulate RT taken to solve addition and subtraction problems, with respect to the number of carries involved in these problems.

Two experiments were conducted in the present study: one on human participants and the other on connectionist models. Both experiments had *learning* and *solving* phases (Figure 1.1). In the learning phase of the human experiment, participants were taught a method for solving binary arithmetic problems by following guiding examples. In the solving phase, participants began the experiment in earnest, solving arithmetic problems under experimental conditions and having their RTs recorded as a measure of problem difficulty. In the learning phase of the model experiment, connectionist models were trained until they achieved 100% accuracy across all problems. I consider this to be roughly

equivalent to how participants were taught to solve arithmetic problems in the learning phase of the human experiment. In the solving phase, all problems were solved again and the number of computational steps taken to solve each problem were recorded as a measure of problem difficulty. Following both experiments, results were analyzed in order to investigate whether any similarities could be observed in how both agents underwent problem difficulty with respect to the number of carries. I then investigated how major model configurations affect model behavior.



## Chapter 2

### Problem Sets

Addition dataset (n=256)					Subtraction dataset (n=136)			
0-carry dataset (n=81)	1-carry dataset (n=54)	2-carry dataset (n=54)	3-carry dataset (n=42)	4-carry dataset (n=27)	0-carry dataset (n=81)	1-carry dataset (n=27)	2-carry dataset (n=19)	3-carry Dataset (n=9)
Addition problem set (n=50)					Subtraction problem set (n=40)			
0-carry problem set (n=10)	1-carry problem set (n=10)	2-carry problem set (n=10)	3-carry problem set (n=10)	4-carry problem set (n=10)	0-carry problem set (n=10)	1-carry problem set (n=10)	2-carry problem set (n=10)	3-carry problem set (n=10)

Figure 2.1: Problem sets. The addition and subtraction datasets were assigned to connectionist models. The addition and subtraction problem sets were assigned to participants.  $n$  refers to the number of operations in a given dataset/problem set.

#### 2.1 Operation Datasets

For addition and subtraction, I constructed separate *operation datasets*, containing all possible operations between two 4-digit binary nonnegative integers that generate nonnegative results. The addition dataset has 256 operations, and the subtraction dataset has 136 operations (Figure 2.1). Operation datasets consist of  $(\mathbf{x}, \mathbf{y})$  where  $\mathbf{x}$  is an 8-dimensional input vector that is a concatenation of two binary operands, and  $\mathbf{y}$  is an output vector that is the result of computing these operands.  $\mathbf{y}$  is 5-dimensional for addition and 4-dimensional for subtraction.

## 2.2 Carry Datasets

Operation datasets were further subdivided into carry datasets. A *carry dataset* refers to the total set of operations in which a specific number of carries is required for a given operator. The addition dataset was divided into 5 carry datasets, and the subtraction dataset was divided into 4 carry datasets (Figure 2.1). For example, in Figure 3.1, the addition guiding examples (a) and (b) are in 2-carry<sup>1</sup> and 4-carry datasets, respectively; the subtraction guiding examples (c) and (d) are in 2-carry and 3-carry datasets, respectively.

---

<sup>1</sup>Let us simply refer to the carry dataset involving  $n$  carries as the  $n$ -carry dataset, and problems from the  $n$ -carry dataset as  $n$ -carry problems.

## Chapter 3

# Experiment 1: Humans

Experiment 1 investigated whether human RT in problem solving increases as a function of the number of carries involved in a problem.

### 3.1 Participants

90 undergraduate and graduate students (48 men, 42 women) from various departments completed the experiment. The average age of participants was 23.6 ( $SD = 3.3$ ).

### 3.2 Materials

Participants were given two types of problem sets: addition and subtraction. The addition problem set was constructed as follows: 10 different problems were sampled from each carry dataset without replacement<sup>1</sup>. These sampled problems were shuffled together to make the addition problem set. This addition problem set was comprised of 50 unique problems evenly distributed across 5 carry datasets (Figure 2.1). Likewise, the subtraction problem sets consisted of 40 problems evenly distributed across 4 carry datasets (Figure 2.1). The problems were newly sampled for each participant.

---

<sup>1</sup> This only occurred when sampling 3-carry problems ( $n = 10$ ) from the 3-carry subtraction dataset ( $n = 9$ ). This required one random problem to be duplicated and shown twice in the 3-carry problem set.

### 3.3 Procedure and Instruments

Participants were shown calculation guidelines containing two guiding examples for addition (Figure 3.1a, 3.1b). Participants were explicitly requested to solve problems by using carry operations outlined in the examples. Participants then began to solve each problem from their addition problem set. The first 5 problems<sup>2</sup>, each of which involved a different number of carry operations, were given sequentially in order to allow participants to practice carry operations and to get used to the experiment interface (Figure 3.2). For each problem, participants followed the procedure as illustrated in Figure 3.2a. In any given problem, two operands were presented in a fixed 4-digit format in order to control for possible extraneous influences on problem difficulty (Ashcraft, 1992, 1995), as outlined by criterion (b). The experiment was designed in such a way that participants were required to click all digits when answering questions (e.g., if the answer was 1, participants were forced to respond with 0001 as opposed to just 1). This was to ensure RTs were not affected by the number of answer digits. The measurement of RTs started as soon as a problem appeared on the screen and stopped when the participant clicked the submission button with all answer digits selected. Measured RTs were accurate to the nearest millisecond. After solving all addition problems, participants repeated the previous procedure for their subtraction problem set (Figure 3.2b) with two subtraction guiding examples (Figure 3.1c, 3.1d). Participants were prohibited from using any writing apparatus in order to force participants to solve problems mentally.

---

<sup>2</sup>4 problems when solving a subtraction problem set

10100 Carry	11110 Carry	0120 Carry	0112 Carry
1011	1111	1001	1000
+ 1010	+ 1011	- 0010	- 0101
10101	11010	0111	0011
(a)	(b)	(c)	(d)

Figure 3.1: Guiding examples

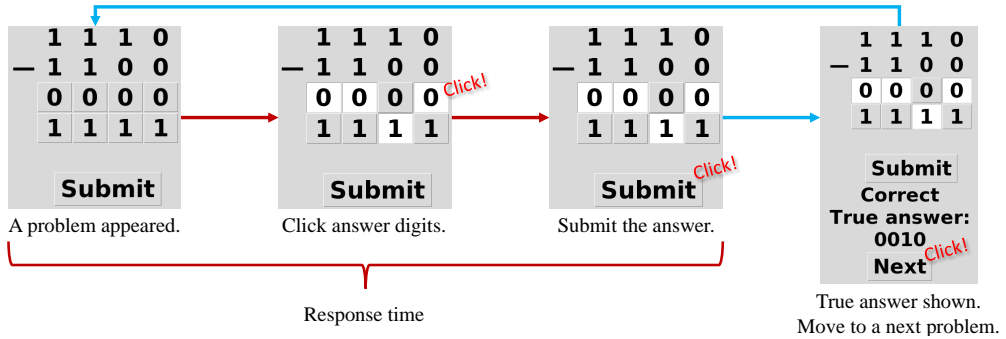
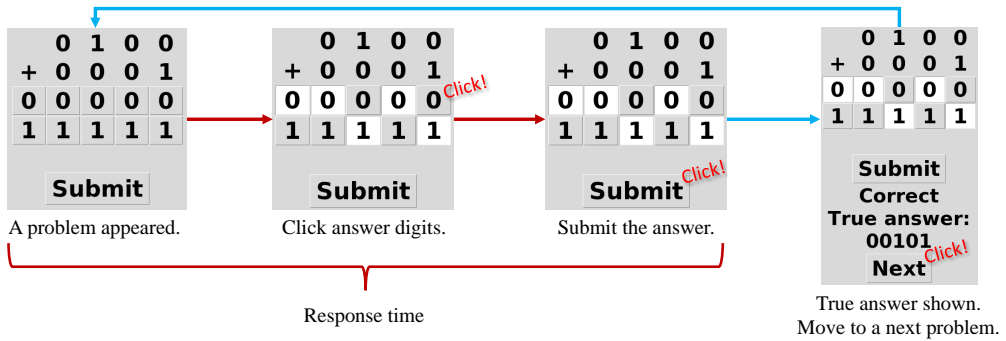


Figure 3.2: Procedure of solving a problem in Experiment 1. Every answer digit should be selected to submit an answer. The number of buttons for answer digits was determined based on the fact that the maximum number of answer digits is 5 for 4-digit addition and 4 for 4-digit subtraction.

## 3.4 Results

Analysis of variance (ANOVA) was used to investigate differences in mean RTs of participants across carry problem sets. If there were significant differences between all the mean RTs, post hoc analysis was applied. If a participant provided a wrong answer, it was reasonable to assume that this participant made some cognitive error when solving the problem. As such, only RTs for correct answers were included in analysis. I removed the outlying RTs of each carry problem set for each participant since unusually short RTs may be due to memory retrieval and excessively long RTs may be caused by distraction or anxiety during problem solving. The RTs in the range  $[Q_1 - 1.5 \cdot \text{IQR}, Q_3 + 1.5 \cdot \text{IQR}]$  were considered outliers, where  $Q_1$  and  $Q_3$  were the first and third quantiles of the RTs for a carry problem set, and  $\text{IQR} = Q_3 - Q_1$ .

### 3.4.1 Addition

There were significant differences in mean RTs between all carry problem sets, as determined by ANOVA [ $F(4, 445) = 51.84, p < .001, \eta^2 = .32$ ]. Post hoc comparisons using the Games-Howell test indicated that mean RTs between any two carry problem sets showed a significant difference [3-carry and 4-carry problem sets:  $p = .040$ ; other pairs:  $p < .01$ ]. Therefore, the mean RT was strictly increasing<sup>3</sup> with respect to the number of carries (Figure 3.3a).

### 3.4.2 Subtraction

There were significant differences in mean RTs between all carry problem sets, as determined by ANOVA [ $F(3, 356) = 117.41, \eta^2 = .50$ ]. Post hoc comparisons using the Games-Howell test indicated that mean RTs between any two carry problem sets showed a significant difference [ $p < .001$ ]. Therefore, the mean RT

---

<sup>3</sup>For every  $x$  and  $x'$  such that  $x < x'$ , if  $f(x) < f(x')$ , then we say  $f$  is *strictly increasing*.

was strictly increasing with respect to the number of carries (Figure 3.3b).

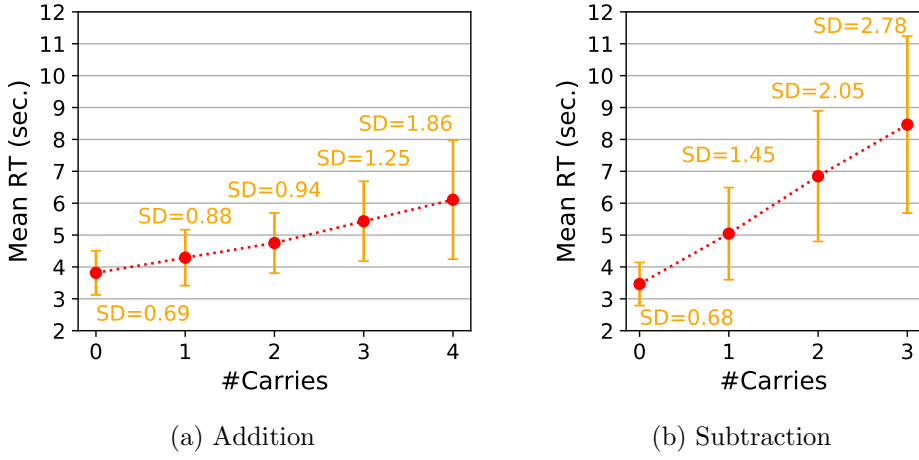


Figure 3.3: Mean RT by carries. The error bars are  $\pm 1SD$ .

Table 3.1: Means (and standard deviations) of mean RTs in Experiment 1

Operator	Carries				
	0	1	2	3	4
<b>Addition</b>	3.81 (0.69)	4.29 (0.88)	4.75 (0.94)	5.43 (1.25)	6.11 (1.86)
<b>Subtraction</b>	3.46 (0.68)	5.04 (1.45)	6.85 (2.05)	8.46 (2.78)	

( $n = 90$  for each group)

## Chapter 4

# Experiment 2: Connectionist Models

Experiment 2 investigated whether computational steps required by connectionist models in problem solving increase as a function of the number of carries involved in a problem. Moreover, this experiment intended to examine how the central model hyperparameters — confidence threshold and hidden dimension — affect the simulated RT. The hidden dimension, denoted by  $d_h$ , refers to the number of units in the hidden layer.

### 4.1 Model

Imagine the human cognitive process while performing addition and subtraction. Humans predict answer digits one by one while mentally referencing two operands and previously predicted digits. Therefore, I aimed to simulate this human cognitive process by using the Jordan network (Jordan, 1997). The Jordan network is a recurrent neural network whose hidden layer gets its inputs from an input at the current step and from the output at the previous step (Figure 4.1). When solving arithmetic problems, humans sometimes follow their automaticity instead of the standard algorithm. Therefore, in order to faithfully simulate this human process, the network was not forced to predict answer digits in one left-to-right or right-to-left sequential direction. These non-sequential predictions let the network learn either addition or subtraction using the same



model structure. This allows for valid comparisons to be drawn between the networks learning addition and the networks learning subtraction.

The Jordan network solves problems as follows: An 8-dimensional input vector  $\mathbf{x}^{(t)}$  composed of two concatenated 4-digit operands is fed into the network (Figure 4.1a). At the same time, its hidden layer  $\mathbf{h}^{(t)}$  with rectifier linear units (ReLU) gets its previous probability outputs  $\mathbf{p}^{(t-1)}$ . The network predicts step-by-step the probabilities of answer digits up to a maximum of 30 steps (Figure 4.1b). At the initial step, all digit predictions are initialized as 0.5, which mimics the initial uncertainty humans experience when solving problems. The output layer gets activated through sigmoid  $\sigma$ . Each output unit predicts each output digit. The network outputs 5-dimensional and 4-dimensional vectors for addition and subtraction problems respectively. The Jordan network used in the present study is formulated as follows:

$$\begin{aligned}\mathbf{h}^{(t)} &= \text{ReLU} \left( W_{\mathbf{x}}^{[1]} \mathbf{x}^{(t)} + W_{\mathbf{p}}^{[1]} \mathbf{p}^{(t-1)} + \mathbf{b}^{[1]} \right) \\ \mathbf{p}^{(t)} &= \sigma \left( W_{\mathbf{x}}^{[2]} \mathbf{x}^{(t)} + W_{\mathbf{p}}^{[2]} \mathbf{p}^{(t-1)} + \mathbf{b}^{[2]} \right)\end{aligned}$$

where

$$t = 0, 1, 2, \dots, 29$$

$$\mathbf{p}^{(0)} = [0.5 \dots 0.5]$$

$$\text{ReLU}(x) = \max(x, 0)$$

$$\sigma(x) = \frac{e^{-x}}{1 + e^{-x}}$$

At each time step, the network predicts the probability of every answer digit. When problem solving, humans only decide on an answer digit when they are sufficiently confident that it is correct. Likewise, the network decides each digit only when its predicted probability  $p_i$  is higher than some threshold. We

call this threshold the *confidence threshold*, denoted by  $\theta_c$ . Suppose  $\theta_c = 0.9$ . If a predicted probability  $p_i$  is in the range  $[0.1, 0.9]$ , the model is *uncertain* about the digit. Otherwise, it is *confident* about the digit: if  $p_i \in [0, 0.1)$ , it predicts the digit is 0; if  $p_i \in (0.9, 1]$ , it predicts the digit is 1. The network is designed to give an *answer* when it is first confident about all answer digits (Figure 4.1b, Algorithm 1). The network in Figure 4.1b answers at step 1 because this is the first state where the model is confident about all digits. At this answer step, the answer is marked as either correct or incorrect. No answer is given if 30 steps are exceeded (Algorithm 1).

---

**Algorithm 1:** How the model answers a problem

---

**Result:** The model's answer is  $\mathbf{z}^{(t)}$  if  $\mathbf{z}^{(t)}$  has been returned. Otherwise, the model does not answer.

```

1 for  $t \leftarrow 0$  to 29 by 1 do
2   Compute  $\mathbf{p}^{(t)}$ ;
3   if  $p_i^{(t)} \in [0, 0.1)$  then
4      $z_i^{(t)} = 0$ ;
5   if  $p_i^{(t)} \in (0.9, 1]$  then
6      $z_i^{(t)} = 1$ ;
7   if  $p_i^{(t)} \in [0, 0.1) \cup (0.9, 1]$  for all  $p_i^{(t)}$  then
8     Return an answer  $\mathbf{z}^{(t)}$ ;
9     break;
10 end
```

---

The network learned arithmetic by minimizing the sum of the losses at all steps  $\sum_t H(\mathbf{z}^{(t)}, \mathbf{p}^{(t)})$  with the backpropagation algorithm (Rumelhart, Hinton, & Williams, 1986). At each step  $t$ , a loss is defined as the cross-entropy  $H$

between the true answer  $\mathbf{z}^{(t)}$  and the output probability vector  $\mathbf{p}^{(t)}$ :

$$H(\mathbf{z}^{(t)}, \mathbf{p}^{(t)}) = -\mathbf{z}^{(t)} \cdot \log \mathbf{p}^{(t)} - (1 - \mathbf{z}^{(t)}) \cdot [1 - \log \mathbf{p}^{(t)}]$$

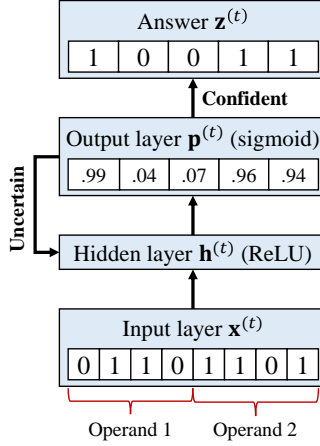
## 4.2 Measures

### 4.2.1 Accuracy

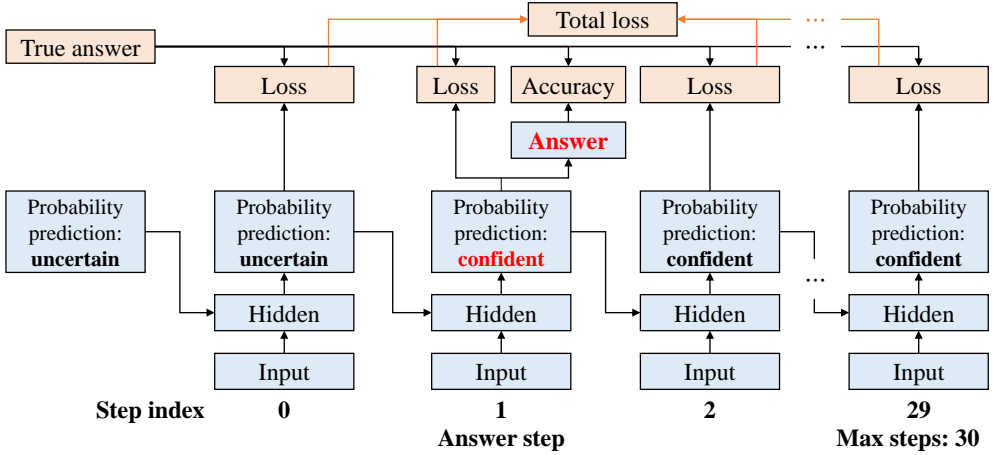
*Accuracy* was measured by dividing the number of correct answers by the total number of problems. Model accuracy was used to measure how successfully the model learned arithmetic and to determine when to stop training. No answer after 30 time steps was considered a wrong answer.

### 4.2.2 Answer Step

*Answer step* was defined as the index of a certain time step where the network outputs an answer. Answer step is roughly equivalent to human RT. It refers to the number of computational steps required for the network to solve an arithmetic problem. Answer step ranges from 0 to 29.



(a) The Jordan network for addition



(b) The Jordan network unrolled through time steps.

Figure 4.1: The Jordan network used in the present study. (a) The network is predicting the answer of  $110 + 1101$  to be  $10011$ . In this example, the confidence threshold is  $0.9$ . At the current state  $t$ ,  $\mathbf{x}^{(t)} = (0, 1, 1, 0, 1, 1, 0, 1)$ ,  $\mathbf{p}^{(t)} = (.99, .04, .07, .96, .94)$ , and  $\mathbf{z}^{(t)} = (1, 0, 0, 1, 1)$ . (b) The network is constrained to compute at most 30 steps. The initial probabilities of answer digits are  $0.5$ , meaning the network is uncertain about all digits. The network repeatedly computes the probabilities of answer digits until it becomes confident about all answer digits; in this figure, it answers at step 1. In the learning phase, the network learns from the total loss from all steps. Accuracy is computed by comparing predicted answers to true answers.

### 4.3 Training Settings

The network learned arithmetic operations by using backpropagation through time (Rumelhart et al., 1986; Werbos, 1990) and a stochastic gradient method (Bottou, 1998) called Adam optimization (Kingma & Ba, 2015) with settings ( $\alpha = .001$ ,  $\beta_1 = .9$ ,  $\beta_2 = .999$ ,  $\epsilon = 10^{-8}$ ). For each epoch, 32-sized mini-batches were randomly sampled without replacement (Shamir, 2016) from the total operation dataset. The weight matrix  $W^{[l]}$  in layer  $l$  was initialized to samples from the truncated normal distribution ranging  $[-1/\sqrt{n^{[l-1]}}, 1/\sqrt{n^{[l-1]}}]$  where  $n^{[l]}$  was the number of units in the  $l$ -th layer; All bias vectors  $b^{[l]}$  were initialized to 0. After training each epoch, accuracy was evaluated on the operation dataset (Figure 2.1). When the network attained 100% accuracy for the entirety of the operation dataset, training was stopped. 300 Jordan networks were trained for each model configuration in order to draw statistically meaningful results. Furthermore, to investigate if any statistically significant relationship held for various model configurations, I reanalyzed the models with the confidence thresholds  $\theta_c \in \{.7, .8, .9\}$  and hidden dimensions  $d_h \in \{24, 48, 72\}$ . 9 types of networks were trained for both addition and subtraction, respectively; a total of 5400 networks were trained in this experiment. I implemented all networks and learning algorithms in Tensorflow (Abadi et al., 2016).

### 4.4 Results

Our proposed model successfully learned all possible addition and subtraction operations between 4-digit binary numbers. The model required 4000 epochs on average (58 minutes<sup>1</sup>) to learn addition, and 1080 epochs on average (13 minutes) to learn subtraction. When training was completed, I examined: (1)

---

<sup>1</sup>Two Intel(R) Xeon(R) CPU E5-2695 v4 and five TITAN Xp were used. Training networks in parallel is vital in this experiment.

statistical differences in mean answer steps between carry datasets across all model configurations; (2) statistical differences in mean answer steps for operation datasets between different confidence thresholds and hidden dimensions.

#### 4.4.1 Addition

The first analysis was conducted on mean answer steps per carry dataset. For every model configuration, ANOVA found significant differences in mean answer steps between all carry datasets (Table 4.2). Post hoc Games-Howell testing found that for 8 of the 9 model configurations, mean answer step was strictly increasing with respect to the number of carries (Table 4.2, Figure 4.2a); the remaining model configuration ( $\theta_c = 0.7$ ,  $d_h = 24$ ) showed a monotonically<sup>2</sup> increasing relationship between mean answer step and the number of carries (Table 4.2).

The second analyses were conducted on mean answer steps for the addition dataset. For every hidden dimension, ANOVA found significant differences in mean answer steps between all confidence thresholds  $\forall \theta_c \in \{.7, .8, .9\}$  (Table 4.3). Post hoc Games-Howell testing found that for all models, mean answer step was strictly increasing with respect to confidence threshold (Table 4.3, Figure 4.3a). For every confidence threshold, ANOVA found significant differences in mean answer steps between all hidden dimensions  $\forall d_h \in \{24, 48, 72\}$  (Table 4.4). Post hoc Games-Howell testing found that with  $\theta_c = 0.7$ , mean answer step was monotonically increasing with respect to hidden dimension. For both other confidence thresholds, mean answer step was strictly increasing with respect to hidden dimension (Table 4.4, Figure 4.4a). We should note however that while significant, the effect of hidden dimension on mean answer step was small.

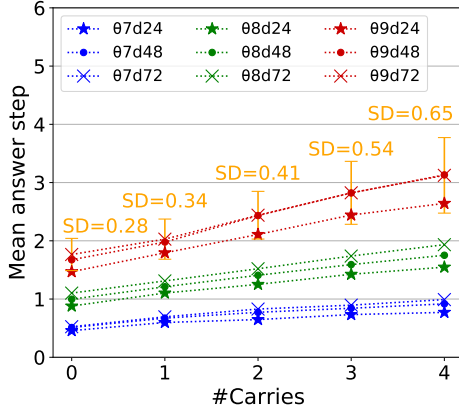
---

<sup>2</sup>For every  $x$  and  $x'$  such that  $x < x'$ , if  $f(x) \leq f(x')$ , then we say  $f$  is *monotonically increasing*.

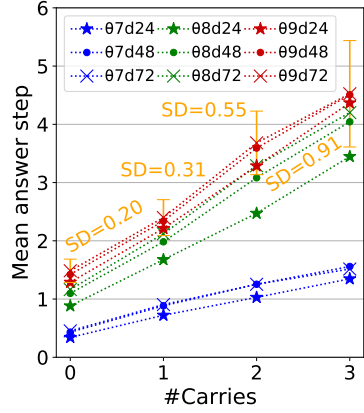
### 4.4.2 Subtraction

The first analysis was conducted on mean answer steps per carry dataset. For every model configuration, ANOVA found significant differences in mean answer steps between all carry datasets (Table 4.2). Post hoc Games-Howell testing found that for all model types, mean answer step was strictly increasing with respect to the number of carries (Table 4.2, Figure 4.2b).

The second analyses were conducted on mean answer steps for the subtraction dataset. For every hidden dimension, ANOVA found significant differences in mean answer steps between all confidence thresholds  $\forall \theta_c \in \{.7, .8, .9\}$  (Table 4.3). Post hoc Games-Howell testing found that for all models, mean answer step was strictly increasing with respect to confidence threshold (Table 4.3, Figure 4.3b). For every confidence threshold, ANOVA found significant differences in mean answer steps between all hidden dimensions  $\forall d_h \in \{24, 48, 72\}$  (Table 4.4). Post hoc Games-Howell testing found that with  $\theta_c = 0.9$ , mean answer step was monotonically increasing with respect to hidden dimension. For both other confidence thresholds, mean answer step was strictly increasing with respect to hidden dimension (Table 4.4, Figure 4.4a). We should note however that while significant, the effect of hidden dimension on mean answer step was small (Figure 4.4a).

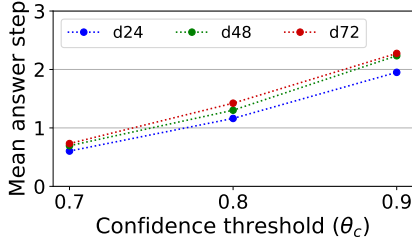


(a) Addition

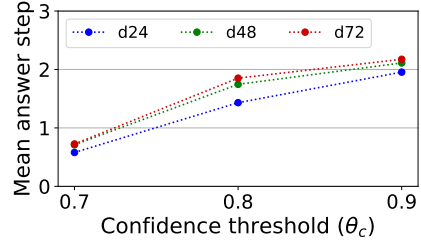


(b) Subtraction

Figure 4.2: Mean answer step by carries (for carry datasets).  $\theta 9d72$  denotes models with  $\theta_c = 0.9$  and  $d_h = 72$ . The error bars are  $\pm 1SD$  and belong to  $\theta 9d72$ .

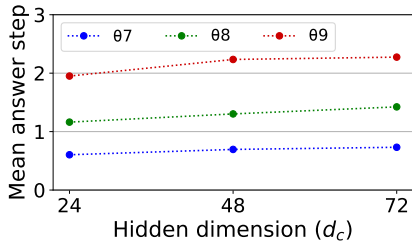


(a) Addition

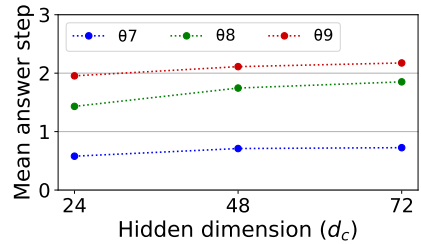


(b) Subtraction

Figure 4.3: Mean answer step by confidence threshold (for operation datasets)



(a) Addition



(b) Subtraction

Figure 4.4: Mean answer step by hidden dimension (for operation datasets)



Table 4.1: Means (and standard deviations) of mean answer steps in Experiment 2

Operator	$\theta_c$	$n_h$	All	Carries				
				0	1	2	3	4
Addition	.7	24	0.60 (0.23)	0.46 (0.25)	0.60 (0.28)	0.65 (0.25)	0.73 (0.24)	0.77 (0.24)
	.7	48	0.70 (0.16)	0.51 (0.19)	0.67 (0.19)	0.77 (0.18)	0.84 (0.17)	0.92 (0.21)
	.7	72	0.73 (0.16)	0.52 (0.20)	0.69 (0.19)	0.83 (0.16)	0.89 (0.16)	0.99 (0.20)
	.8	24	1.16 (0.36)	0.88 (0.38)	1.10 (0.40)	1.25 (0.37)	1.43 (0.38)	1.55 (0.47)
	.8	48	1.30 (0.32)	1.00 (0.32)	1.21 (0.31)	1.41 (0.34)	1.59 (0.38)	1.75 (0.47)
	.8	72	1.42 (0.41)	1.10 (0.33)	1.31 (0.34)	1.52 (0.45)	1.74 (0.55)	1.93 (0.64)
	.9	24	1.95 (0.47)	1.47 (0.45)	1.79 (0.47)	2.11 (0.51)	2.44 (0.63)	2.64 (0.74)
	.9	48	2.23 (0.38)	1.67 (0.34)	1.98 (0.38)	2.43 (0.46)	2.82 (0.60)	3.13 (0.66)
	.9	72	2.27 (0.34)	1.76 (0.28)	2.03 (0.34)	2.44 (0.41)	2.82 (0.54)	3.12 (0.65)
Subtraction	.7	24	0.58 (0.21)	0.34 (0.17)	0.72 (0.27)	1.03 (0.35)	1.35 (0.47)	
	.7	48	0.71 (0.18)	0.43 (0.14)	0.88 (0.22)	1.25 (0.32)	1.56 (0.45)	
	.7	72	0.73 (0.18)	0.45 (0.14)	0.90 (0.21)	1.25 (0.28)	1.52 (0.46)	
	.8	24	1.43 (0.44)	0.88 (0.29)	1.68 (0.50)	2.47 (0.86)	3.45 (1.62)	
	.8	48	1.75 (0.42)	1.10 (0.26)	1.98 (0.45)	3.08 (0.93)	4.04 (1.52)	
	.8	72	1.85 (0.42)	1.16 (0.26)	2.12 (0.47)	3.29 (0.92)	4.20 (1.42)	
	.9	24	1.95 (0.36)	1.29 (0.32)	2.21 (0.48)	3.29 (0.67)	4.37 (1.19)	
	.9	48	2.11 (0.25)	1.42 (0.24)	2.34 (0.33)	3.59 (0.55)	4.51 (0.89)	
	.9	72	2.17 (0.24)	1.49 (0.20)	2.40 (0.31)	3.68 (0.55)	4.52 (0.91)	

( $n = 300$  for each group)

Table 4.2: The results of ANOVA and post hoc analysis on differences in mean answer steps between all carry datasets. The model configuration varies along two axes: confidence threshold and hidden dimension. 300 mean answer steps per carry dataset from 300 trained networks were analyzed for each model configuration.  $F$  is the  $F$ -test statistic and  $\eta^2$  is the effect size from ANOVA; in addition, there were 4 degrees of freedom between carry datasets and 1495 within carry datasets:  $df_b^+ = 4$ ,  $df_w^+ = 1495$ ; in subtraction,  $df_b^- = 3$ ,  $df_w^- = 1196$ . The mean answer step columns describe the results of post hoc analysis. The inequality ( $<$ ) denotes a significant difference at the  $p < .05$  level. Equality ( $=$ ) denotes the opposite. The numbers in these columns refer to the number of carries of a carry dataset. \*  $p < .05$ . \*\*  $p < .01$ . \*\*\*  $p < .001$ .

		Addition			Subtraction		
$\theta_c$	$d_h$	$F$	$\eta^2$	Mean answer step	$F$	$\eta^2$	Mean answer step
.7	24	72***	.16	$0 < 1 = 2 < 3 = 4$ ***	499***	.56	$0 < 1 < 2 < 3$ ***
.7	48	206***	.36	$0 < 1 < 2 < 3 < 4$ ***	765***	.66	$0 < 1 < 2 < 3$ ***
.7	72	294***	.44	$0 < 1 < 2 < 3 < 4$ ***	716***	.64	$0 < 1 < 2 < 3$ ***
.8	24	129***	.26	$0 < 1 < 2 < 3 < 4$ **	390***	.49	$0 < 1 < 2 < 3$ ***
.8	48	198***	.35	$0 < 1 < 2 < 3 < 4$ ***	571***	.59	$0 < 1 < 2 < 3$ ***
.8	72	142***	.28	$0 < 1 < 2 < 3 < 4$ **	674***	.63	$0 < 1 < 2 < 3$ ***
.9	24	208***	.36	$0 < 1 < 2 < 3 < 4$ **	970***	.71	$0 < 1 < 2 < 3$ ***
.9	48	421***	.53	$0 < 1 < 2 < 3 < 4$ ***	1769***	.82	$0 < 1 < 2 < 3$ ***
.9	72	432***	.54	$0 < 1 < 2 < 3 < 4$ ***	1718***	.81	$0 < 1 < 2 < 3$ ***

Table 4.3: The results of ANOVA and post hoc analysis on differences in mean answer steps between confidence thresholds.  $df_b^+ = df_b^- = 2$ .  $df_w^+ = df_w^- = 897$ . In the mean answer step columns, the numbers refer to confidence thresholds.

$d_h$	Addition			Subtraction		
	$F$	$\eta^2$	Mean answer step	$F$	$\eta^2$	Mean answer step
24	1032***	.70	.7 < .8 < .9***	1163***	.72	.7 < .8 < .9***
48	2002***	.82	.7 < .8 < .9***	1736***	.79	.7 < .8 < .9***
72	1735***	.79	.7 < .8 < .9***	1963***	.81	.7 < .8 < .9***

Table 4.4: The results of ANOVA and post hoc analysis on differences in mean answer steps between hidden dimensions.  $df_b^+ = df_b^- = 2$ .  $df_w^+ = df_w^- = 897$ . In the mean answer step columns, the numbers refer to hidden dimension.

$\theta_c$	Addition			Subtraction		
	$F$	$\eta^2$	Mean answer step	$F$	$\eta^2$	Mean answer step
.7	58***	.08	24 < 48 = 72***	46***	.10	24 < 48 < 72**
.8	38***	.08	24 < 48 < 72***	77***	.15	24 < 48 < 72**
.9	37***	.12	24 < 48 < 72*	51***	.09	24 < 48 = 72***

## Chapter 5

# Discussion and Conclusion

**Experiment 1** Experiment 1 has improved the previous study (Cho, Lim, Hickey, & Zhang, 2019) as follows: Firstly, participants were forced to solve problems using solely mental arithmetic. This allows for more valid comparisons to be drawn between humans and models. Secondly, larger data samples allowed the present study to find more statistically significant results. Specifically, mean RT for addition problems were found to be strictly increasing with respect to the number of carries.

**Experiment 2** In Experiment 2, the two hyperparameters — confidence threshold and hidden dimension — were chosen since I expected these hyperparameters to correspond to humans’ uncertainty and memory capacity, respectively. I further expected that increasing confidence threshold and decreasing hidden dimension would increase answer step. This expectation subsequently arose for confidence threshold; confidence threshold had an augmenting effect on answer step. However, my expectation was not born out for hidden dimension. In order to observe clear differences in mean answer steps with respect to problem difficulty, high confidence thresholds are recommended. Hidden dimension should be fixed to the extent that the model can learn an entire dataset.

The proposed Jordan network is distinct from neural networks previously

studied from the following perspectives: First, the Jordan network learns based on the gradient descent algorithm rather than the Hebbian learning used by associative-memory networks (Anderson et al., 2004; McCloskey & Lindemann, 1992; Viscuso et al., 1989). Second, the Jordan network is able to perform multi-digit addition and subtraction, rather than single-digit operations used in previous studies (Anderson et al., 2004; Mickey & McClelland, 2014; McCloskey & Lindemann, 1992; Viscuso et al., 1989). Finally, my proposed model utilizes computational steps to simulate human RT, while the NeuralGPU model (Kaiser & Sutskever, 2016) does not (even though NeuralGPU correctly captures the concept of carry operations).

**Experiments 1 & 2** The preceding results show three notable similarities between humans and my connectionist models: Firstly, both agents experienced increased levels of difficulty as more carries were involved in arithmetic problems. Secondly, the Jordan networks with the model configuration ( $\theta_c = 0.9$ ,  $d_h = 72$ ) successfully mimicked the increasing standard deviation of human RT with respect to the number of carries (Figure 3.3, 4.2). This phenomenon could not be achieved by a rule-based system performing the standard algorithm, although such a system would be able to simulate increasing RT as a function of the number of carries. Lastly, another similarity found between both humans and models is that the difficulty slope for subtraction is steeper than for addition (Figure 3.3, 4.2). This implies that the augmenting effect of carries on problem difficulty is stronger in subtraction than in addition.

**Contributions** The present study makes two major contributions to the literature: Firstly, my models successfully simulated humans’ RT in terms of these three similarities: increasing latency, increasing standard deviation of latency,

and relative steepness of increasing latency. The similarities may suggest that some cognitive process, equivalent to the nonlinear computational process used in the Jordan network, could be involved in human cognitive arithmetic. Secondly, the present study demonstrated that fitting my model to arithmetic data induced human-like latency to emerge in the connectionist models (McClelland et al., 2010). In other words, human RTs to arithmetic problems were successfully learned in an unsupervised way. This contrasts with previous studies that focus on learning arithmetic tasks in a supervised way.

**Future Study** The present study focuses solely on analyzing mean answer steps between arithmetic problem sets of varying difficulty levels. Therefore, future studies could aim to better understand what dynamic processes my model uses when solving individual problems: Specifically, it might be interesting to observe how my model predicts individual digits through each time step when solving problems. Also, it may be worth adding attention mechanisms (Bahdanau, Cho, & Bengio, 2015; Vaswani et al., 2017) to the proposed Jordan network, in order to imitate humans’ selective attention on operands while performing arithmetic. Furthermore, similarities between both the model’s sequentially predictive answering process and the human answering process could be investigated. This comparison would give us a better understanding of both my model and human mathematical cognition (McClelland et al., 2016).

My model is designed not just for arithmetic cognition, but also for sequential predictions that based on a constant input and a previous prediction, which result in a single answer. In this regard, this model has the potential to be applied to other cognitive processes involving sequential processing and RT as a measure of cognitive difficulty. Therefore, future studies could consider extending my model to other domains of cognition. For example, well

known character image and word classification datasets can be subdivided into datasets of varying difficulty levels, similar to my carry datasets. Mean answer steps for classifying these data sets could be analyzed using a similar model to that outlined in the present study.

# References

- Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., . . . Zheng, X. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, *abs/1603.04467*.
- Anderson, J. A., Spoehr, K. T., & Bennett, D. J. (2004). A study in numerical perversity: Teaching arithmetic to a neural network. In D. S. Levine & M. Aparicio (Eds.), *Neural networks for knowledge representation and inference* (pp. 311–335). Hillsdale, NJ: Lawrence Erlbaum Associates.
- Ashcraft, M. H. (1992). Cognitive arithmetic: A review of data and theory. *Cognition*, *44*, 75–106.
- Ashcraft, M. H. (1995). Cognitive psychology and simple arithmetic: A review and summary of new directions. *Mathematical Cognition*, *1*(1), 3–34.
- Baddeley, A. D., & Della Sala, S. (1996). Working memory and executive control. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, *351*(1346), 1397–1404.
- Baddeley, A. D., & Hitch, G. (1974). Working memory. In G. H. Bower (Ed.), *Psychology of learning and motivation* (Vol. 8, pp. 47–89). Academic Press.
- Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *3rd international conference on learning representations*. Retrieved from <http://arxiv.org/abs/1409.0473>
- Bengio, Y., Courville, A. C., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, *35*(8), 1798–1828.
- Bottou, L. (1998). Online algorithms and stochastic approximations. In D. Saad (Ed.), *Online learning and neural networks*. Cambridge, UK: Cambridge University Press.
- Campbell, J. I. (1994). Architectures for numerical cognition. *Cognition*, *53*(1), 1–44.
- Chen, S., Zhou, Z., Fang, M., & McClelland, J. (2018). Can generic neural networks estimate numerosity like humans? In *Proceedings of the 40th annual meeting of the Cognitive Science Society* (pp. 202–207).



- Cho, S., Lim, J., Hickey, C., & Zhang, B.-T. (2019). Problem difficulty in arithmetic cognition: Humans and connectionist models. In *Proceedings of the 41st annual meeting of the Cognitive Science Society* (pp. 1506–1512).
- Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2), 179–211.
- Fang, M., Zhou, Z., Chen, S., & McClelland, J. (2018). Can a recurrent neural network learn to count things? In *Proceedings of the 40th annual meeting of the Cognitive Science Society* (pp. 360–365).
- Franco, L., & Cannas, S. A. (1998). Solving arithmetic problems using feed-forward neural networks. *Neurocomputing*, 18(1), 61–79.
- Fürst, A. J., & Hitch, G. J. (2000). Separate roles for executive and phonological components of working memory in mental arithmetic. *Memory & Cognition*, 28(5), 774–782.
- Hoshen, Y., & Peleg, S. (2016). Visual learning of arithmetic operation. In *Proceedings of the thirtieth AAAI conference on artificial intelligence* (pp. 3733–3739).
- Imbo, I., Vandierendonck, A., & De Rammelaere, S. (2007). The role of working memory in the carry operation of mental arithmetic: Number and value of the carry. *The Quarterly Journal of Experimental Psychology*, 60(5), 708–731.
- Imbo, I., Vandierendonck, A., & Vergauwe, E. (2007). The role of working memory in carrying and borrowing. *Psychological Research*, 71(4), 467–483.
- Jordan, M. I. (1997). Serial order: A parallel distributed processing approach. In *Advances in psychology* (Vol. 121, pp. 471–495).
- Kaiser, L., & Sutskever, I. (2016). Neural GPUs learn algorithms. In *3rd international conference on learning representations*. Retrieved from <http://arxiv.org/abs/1511.08228>
- Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *2nd international conference on learning representations*. Retrieved from <http://arxiv.org/abs/1412.6980>
- Klein, E., Moeller, K., Dressel, K., Domahs, F., Wood, G., Willmes, K., & Nuerk, H.-C. (2010). To carry or not to carry – is this the question? disentangling the carry effect in multi-digit addition. *Acta Psychologica*, 135(1), 67–76.
- Kuefler, A., Kochenderfer, M. J., & McClelland, J. L. (2017). Geometric concept acquisition in a dueling deep q-network. In *Proceedings of the 39th annual meeting of the Cognitive Science Society* (pp. 2488–2493).
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521,

436–444.

- LeFevre, J.-A., Bisanz, J., Daley, K. E., Buffone, L., Greenham, S. L., & Sadesky, G. S. (1996). Multiple routes to solution of single-digit multiplication problems. *Journal of Experimental Psychology: General*, 125(3), 284–306.
- McClelland, J. L. (1988). Connectionist models and psychological evidence. *Journal of Memory and Language*, 27(2), 107–123.
- McClelland, J. L., Botvinick, M. M., Noelle, D. C., Plaut, D. C., Rogers, T. T., Seidenberg, M. S., & Smith, L. B. (2010). Letting structure emerge: connectionist and dynamical systems approaches to cognition. *Trends in Cognitive Sciences*, 14(8), 348–356.
- McClelland, J. L., Mickey, K., Hansen, S., Yuan, A., & Lu, Q. (2016). A parallel-distributed processing approach to mathematical cognition. *Manuscript, Stanford University*. Retrieved from <https://stanford.edu/~jlmcc/papers/McCEtAl16MsPDPApproachToMathematicalCognition.pdf>
- McCloskey, M., & Lindemann, A. M. (1992). MATHNET: Preliminary results from a distributed model of arithmetic fact retrieval. In J. I. D. Campbell (Ed.), *The nature and origin of mathematical skills* (pp. 365–409). Amsterdam: Elsevier.
- McNeil, N. M. (2007). U-shaped development in math: 7-year-olds outperform 9-year-olds on equivalence problems. *Developmental Psychology*, 43(3), 687–695.
- Mickey, K. W., & McClelland, J. L. (2014). A neural network model of learning mathematical equivalence. In *Proceedings of the 36th annual meeting of the Cognitive Science Society* (pp. 1012–1017).
- Miller, K., Perlmutter, M., & Keating, D. (1984). Cognitive arithmetic: Comparison of operations. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 10(1), 46–60.
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323, 533–536.
- Rumelhart, D. E., & McClelland, J. L. (1986). *Parallel distributed processing* (Vol. 1). MIT Press.
- Saxton, D., Grefenstette, E., Hill, F., & Kohli, P. (2019). Analysing mathematical reasoning abilities of neural models. In *7th international conference on learning representations*. Retrieved from <https://openreview.net/forum?id=H1gR5iR5FX>
- Shamir, O. (2016). Without-replacement sampling for stochastic gradient methods. In *Advances in neural information processing systems 29: Annual conference on neural information processing systems 2016* (pp. 46–54).
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N.,

- ... Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems 30: Annual conference on neural information processing systems 2017* (pp. 6000–6010).
- Viscuso, S. R., Anderson, J. A., & Spoehr, K. T. (1989). Representing simple arithmetic in neural networks. *Advances in Cognitive Science*, 2, 141–164.
- Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10), 1550–1560.
- Zorzi, M., Stoianov, I., & Umiltà, C. (2005). Computational modeling of numerical cognition. In J. I. D. Campbell (Ed.), *Handbook of mathematical cognition* (pp. 67–84). New York: Psychology Press.

## 국문초록

본 연구는 산술 문제를 풀 때 사람과 연결주의 모형이 겪는 어려움이 유사한지를 조사하였다. 문제의 난이도는 주어진 문제를 해결하는데 수반되는 올림의 수에 영향을 받는다. 이 연구는 시간에 따라 동적으로 계산하는 연결주의 모형인 조단 신경망(Jordan network)을 통해, 덧셈 혹은 뺄셈을 푸는 사람의 응답 시간을 모사하고자 하였다. 조단 신경망은 은닉층이 현재 입력값과 이전 예측값을 입력으로 받는 순환 신경망이다. 이 연구에서 문제 난이도를 사람의 응답 시간으로, 모형의 계산 걸음 수로 측정하였다. 연구 결과, 사람과 연결주의 모형 모두가 이진 덧셈과 뺄셈을 풀 때, 올림 수가 증가할수록 어려움을 겪음을 발견하였다. 구체적으로, 두 실험 대상 모두는 올림 수에 따라 문제 난이도가 강한 증가(strictly increasing) 경향을 보였다. 게다가, 문제에 올림 수가 많아질수록 사람이 문제를 푸는데 걸리는 응답 시간의 표준편차가 증가하였는데, 제안한 모형은 그 현상을 모방하였다. 사람과 모형의 또 다른 유사점은 올림 수에 대한 문제 난이도가 덧셈보다 뺄셈에서 더 가파르게 증가했다는 점이었다. 모형의 두 가지 하이퍼 파라미터 — ‘신뢰 임계값’과 ‘은닉 차원’ — 에 대한 추가 조사 결과, 신뢰 임계값이 커질수록 모형이 정답에 도달하기 위해 더 많은 계산 걸음 수를 가지었다. 한편, 은닉 차원이 커질수록 모형이 정답에 도달하기 위해 더 많은 계산 걸음 수를 취했지만, 증가율은 무시할 만한 정도이었다.

**주요어:** 산술 인지; 문제 난이도; 응답 시간; 연결주의 모형; 순환 신경망; 조단 신경망; 계산 걸음 수

**학번:** 2017-28413